Podstawy języka Python:

Zrozumienie podstawowych typów danych (liczby, łańcuchy znaków, listy, słowniki, krotki, zbiory).

Liczby

`int` - liczby całkowite

`float` - liczby zmiennoprzecinkowe

`complex` -liczby zespolone

`a=5 a=3.4 a=3+2,7j`

Stringi - łańcuchy znaków

```
a="string"
```

```
a='string'
```

```
a="string"
```

`a.lower()`-zamienia na małe litery

`a.upper()`-zamienia na wielkie litery

`a.split(",")`-dzieli stringa po ",", zamieia na tablice

`a.strip()`-usuwa białe znaki

`a.isalumn()`-bool, czy jest znakiem alfanumeryczny

`a.isdigit()`-bool, czy jest liczbą

`a.isdecimal()`-bool, czy jest liczbą dziesiętną

`a.isnumeric()`-bool, czy wszystkie znaki są cyframi

`a.index(n)`- podaje n-ty wyraz

`a.count("a")` - liczy ile razy występuje znak "a"

`len(a)`- długość stringa od 1

```
string też jest tablicą, więc
```

`a[0]`- zerowy znak

`a[2:3]`- 3 znak

`a[2:4]`- 3,4 znak

Konwersja

`int(string)` -konwersja stringa na int `str(int)` -konwersja int na string

Listy

```
a=['sds',2,'sd3']
```

`a=[1]`-pierwszy(czyli drugi) wyraz tablicy

`a.append("n")`- dodaje do końca tablicy "n"

`a.remove(1)`-usuwa pierszy napotkany element tablicy ==1

`a.pop()`-usuwa ostatni element

`a.sort()`-sortuje od najmniejszego do najwiekszego

`a.reverse()`-odwraca kolejnosc wyrazów tablicy

`a.clear()`-czyści całą tablicę, tablica pusta

`a.copy()`-kopiuje tablice

`a.count(1)`-zwraca ile wystąpień 1 jest w tablicy

`a.extend(y)`-usuwa z tablicy a elementy zawarte w tablicy y

`enumerate(lista)` -iteruje po indeksie i wartosci tablicy

Słowniki

`dict={}` -tworzenie słownika pustego

`dict={key:value}` -tworzenie słownika

`dict[klucz]=wartosc` -przypisanie wartosci do klucza

`dict.keys()` -lista kluczy

`dict.values()` -lista wartosci slownika

`dict.get(klucz)` -zwraca wartosc

`dict.get(wartosc)` -zwraca klucz

`del slownik[klucz]` -usuwa element ze słownika o kluczu klucz

`dict.items()` -zwraca liste krotek (klucz,wartosc)

`dict.update(dict2)` -dodaje dict2 do słownika dict

Krotki

`tuple=(a,b)`

`tuple[1]` -zwraca 1 element krotki (b)

`tuple.count(a)` -zwraca ilosc elementow a w krotce

`len(tuple)` -zwraca ilosc elementow w krotce

`min(tuple)` -zwraca najmniejsza wartosc w krotce

`max(tuple)` -zwraca najwieksza wartosc w krotce

`tuple(lista)` -konwersja listy lista na krotke

Operacje na danych, takie jak indeksowanie, wycinanie, iteracja.

Indeksowanie

```
lista = [10, 20, 30, 40, 50]
```

`element = lista[2]` -zwraca 2 element listy

Wycinanie

```
lista = [10, 20, 30, 40, 50]
```

`podciag = lista[1:4]` -zwraca elementy 2,3,4 listy

Interacja

```
lista = [10, 20, 30, 40, 50]
for element in lista:
    print(element)
```

Podstawowe konstrukty językowe, w tym instrukcje warunkowe (if, else, elif), pętle (for, while) i wyrażenia listowe (list comprehensions).

If, elif,else

```
if warunek:
    print("cos")
elif warunek3:
    print("cos")
else:
    print('cos')
```

Pętla for

```
for bierzacy_element in zbiór:
    instrukcja_1_pętli_for
    instrukcja_2_pętli_for
    instrukcja_3_pętli_for

for i in range(11):
    print(i)

lista = ["Adam", "Iwona", "Kamila", "Marcin"]
for imie in lista:
    print(imie)
```

Pętla while

```
x = 1
while x <= 100: # dopóki warunek będzie prawdziwy, powtórz poniższe instrukcje
    print(x)    # wypisz wartość zmiennej x
    x = x + 1
```

Wyrażenia listowe (list comprehensions)

```
numbers = [1, 2, 5, 7, 8, 9, 12, 25, 100]
squares = [number ** 2 for number in numbers]
print(squares)
# [1, 4, 25, 49, 64, 81, 144, 625, 10000]
```

Funkcje i moduły:

Tworzenie i używanie funkcji, rozumienie koncepcji zakresu zmiennych.

```
def nazwa_funkcji(argument):
    kod
    return argument * kod
```

Mapowanie

Funkcja `map()` funkcję podaną jako pierwszy argument stosuje do każdego elementu sekwencji podanej jako argument drugi:

```
def kwadrat(x):
    return x**2

kwadraty = map(kwadrat, range(10))
list(kwadraty)
```

Importowanie i wykorzystywanie modułów oraz bibliotek standardowych Pythona.

os

`os.getcwd()` -pokazuje aktualna sciezke

`os.chdir("\\eos\STUDENTs$\334535\Desktop\spi")` -zmiana katalogu roboczego

`os.listdir()` -lista plikow w katalogu w ktorym sie znajdujemy

`os.mkdir("nazwa")` -tworzy katalog

`os.rmdir('nazwa")` -usuwa katalog

`os.remove("nazwa")` -usuwa plik

`os.rename("oldname","nowanazwa")`

`os.path/.dirname/.isdir/.isfine/.exist/("nazwapliku.txt")`

`os.path.split(nazwapliku)` -podzial sceizki pliku na sciezke i nazwe pliku

`os.system("echo Hello Wordl")` -cos jak bash

sys

`sys.argv` -wyciage zmienne srodowiskowe

`sys.path.append("/sciezka)` -dodanie zmiennej

`sys.exit()` -wyjscie z programu

`sys.version` -info o wersji

`sys.version.info` -wiecej info o wersji

`sys.stdout()`

`sys.platform` -informacje o platformie

```
#zapis do pliku 'hello'
with open("output", 'w')as f:
    sys.out=f
    print("hello")
sys.stdout = sys.__stdout__
```

math

`math.sqrt()` -pierwiastek

`math.pi` -wartosc pi

`math.e` -wartosc e

`math.ceil()` -zaokrąglenie w góre

`math.floor()` -zaokrąglenie w dół

`math.gcd()` -najwiekszy wspolny dzielnik

`math.log()` -logarytmy

`math.log10()` -logarytm 10

`math.pow(liczba,potega)` -potegowanie

argparse

```
parser=argparse.ArgumentParser(
    description-"program do wyswietlenia listy plikow"
)
parser.add_argument("echo", help='wyswietlenie stringa')
parser.add_argument('-v',help="wicej danych o output",action='store_true')
args=parser.parse_args()
```

Daty

```
import datetime
import sys
```

```
        #FORMATOWANIE DATY
now = datetime.datetime.now()
#wyswietlanie dzisiejszej daty i godziny
print(f"Dzisiejsza data: {now}")

#wyswietl tylko rok
now_year= now.strftime("%Y")
print(f"dzisiejszy rok: {now_year}")

#wyswietlanie daty w formacie yyyy-mm-dd
data= now.strftime("%Y"'-' "%m"'-' "%d")
print(f"Dzisiejsza data inny format bez godziny {data} ")

#wyswietl tylko miesiąc pełna nazwa
now_month= now.strftime("%B")
print(f"Miesiac pelna nazwa po ang: {now_month}")

#miesiąc skrót nazwy
now_month= now.strftime("%b")
print(f"Miesiac skrócona nazwa po ang: {now_month}")

#miesiąc cyfra
now_month=now.strftime("%m")
print(f"Miesiac cyfra {now_month}")

#miesiąc po polsku
miesiac= {1:"Styczeń",2:"Luty",3:"Marzec",4:"Kwiecień",5:"Maj",6:"Czerwiec",7:"Lipiec",8:"Sierpień",9:"Wrzesień",10:"Październik",11:"Listopad",12:"Grudzień"}
now_month= int(now.strftime("%m"))
month= miesiac[now_month]
print(f"Miesiac po polsku: {month}")

#wyswietl dzien tygodnia
day= now.strftime("%A")
print(f"dzien tygodnia: {day}")

#dodanie n dni, m tygodni do aktualnej daty
add= datetime.datetime.now()
new= add + datetime.timedelta(days=5, weeks=-2)
add= new.strftime("%Y"'-'"%m"'-'"%d")
```

```
print(f"dodane n dni i m tygodni do daty {add}")

#podanie przez uzytkownika daty w formacie dd.mm.yyyy i zamiana na date w formacie yyyy-mm-dd
try:
    today= input("podaj date w formacie dd.mm.yyyy : ")
    today_date = datetime.datetime.strptime(today, '%d'"."'%m'"."'%Y')
    today_new_date= today_date.strftime("%Y""-""%m""-""%d")
    print(f"zmieniony format daty podanej przez uzytkownika: {today_new_date}")

except ValueError:
    print("Podaj prawidlowy format daty")
    sys.exit()

        #RÓŻNICE MIEDZY DATAMI
#roznica w latach miedzy podaną data a dzisiejszą
now= datetime.datetime.now()
now_year= now.year
today_date_year=today_date.year
roznica= now_year - today_date_year
print(f"Róznica w latach od podanej daty wynosi: {roznica} lat")
    #roznica w dniach
now= datetime.datetime.now()
roznica= now - today_date
print(f"Róznica w dniach od podanej daty wynosi: {roznica.days} dni")
    #róznica w tygodniach
now= datetime.datetime.now()
roznica= now - today_date
roznica_weeks= roznica.days/7
print(f"Róznica w tygodniach od podanej daty wynosi: {int(roznica_weeks)} tygodni")
    #róznica w godzianch
now= datetime.datetime.now()
roznica= now - today_date
roznica_hours= roznica.days*24
print(f"Róznica w godzinach od podanej daty wynosi: {roznica_hours} godzin")
    #róznica w minutach
now= datetime.datetime.now()
roznica= now - today_date
roznica_min= roznica.days*24*60
print(f"Róznica w minutach od podanej daty wynosi: {roznica_min} minut")
    #róznica w sekundach
now= datetime.datetime.now()
roznica= now - today_date
roznica_sec= roznica.days*24*60*60
print(f"Róznica w sekundach od podanej daty wynosi: {roznica_sec} sekund")
```

Podstawy programowania obiektowego:

Zrozumienie koncepcji klas i obiektów.

```
#paradygmat programowania- szablon prawidłowego programowania. oop -objective oriented programming. klasa składa sie z obiektow
class Car:
    #atrybuty klasy (dla wszystkich funkcji):
    kolor='czerwony'

    def __init__(self,make,model,year):#dunder metods, konstruktor
        #atrybuty instancji
        self.make=make
        self.model=model
        self.__year=year #atrybut ukryty bo ma __

    def get_year(self): #metoda getter
        return self.__year

    def set_year(self,new_year): #metoda setter
        self.__year=new_year


#metoda=funkcja,    atrybut=zmienna
car=Car('tayota','camry',2023)
# print(car.kolor)
# car.kolor='zielony'
# print(car.kolor)
# car.uszkodzony='bezwypadkowy'   #atrybuty dynamiczne- nie zdefiniowane w klasie
# print(car.uszkodzony)
# car.__year=2020
#print(car.__year)#nie wypisuje bo jest ukryty
```

```
print(car.get_year())
car.set_year(2020)

#DEKORATORY
def my_decorator(func):
    def wrapper():
        print("Tekst przd funkcaj")
        func()
        print("Teskt po wykonaniu funkcji")
    return wrapper

@my_decorator
def czesc():
    print("Hello world")

czesc()

#METODY
class Kwadrat:
    #atrybuty
    width=0
    height=0
    def __init__(self,width,height):
        self.width=width
        self.height=height
    @classmethod    #parametr, dekorator
    def pole_kwadratu(cls,atrybuty):#cls atrybuty klasowe wchodza
        return cls(atrybuty,atrybuty)

    @staticmethod #statyczna metoda, ktora nie pobiera klasy, niepobiera żadnych danych z klas
    def obwod(a,b):
        return 2*a+2*b

class ParrentClass:
    def speak(self):
        print("JESTEM rodzicem")

class ChildClass(ParrentClass):
    def speak(self):
        super().speak()#dziedziczona funkcja
        print("jstem dzieckiem")

child=ChildClass()
child.speak()
parrent=ParrentClass()
parrent.speak()


            #polimorfizm - dziala podobnie
def area(shape):
    return shape.calculate_area()

class Circle:
    def __init__(self,radius):
        self.radius=radius
    def calculate_area(self):
        return 3.14*self.radius**2
class Rectangle:
    def __init__(self,width,height):
        self.width=width
        self.height=height
    def calculate_area(self):
        return self.width*self.height


circle=Circle(4)
rectangle=Rectangle(3,4)
print(f"pole kola= {area(circle)}")
print(f"pole prostokata= {area(rectangle)}")

        #inne rozw.
class Animal:
    def speak(self):
        pass
class Dog:
    def speak(self):
        return "Hau!"
class Cat:
    def speak(self):
        return "Miau!"
```

```
def make_animal_speak(animal):
    return animal.speak()

dog=Dog()
cat=Cat()
print(make_animal_speak(dog))
print(make_animal_speak(cat))




        #klasy abstrakcyjne
from abc import ABC,abstractmethod
class Shape(ABC):
    @abstractmethod
    def calculate_area(shape):
        pass

class Circle(Shape):
    def __init__(self,radius):
        self.radius=radius
    def calculate_area(self):
        return 3.14*self.radius**2
class Rectangle(Shape):
    def __init__(self,width,height):
        self.width=width
        self.height=height
    def calculate_area(self):
        return self.width*self.height


circle=Circle(4)
rectangle=Rectangle(3,4)
print(f"pole kola= {circle.calculate_area()}")
print(f"pole prostokata= {rectangle.calculate_area()}")



            #magiczne metody
class Person:
    def __init__(self,name,age):
        self.name=name
        self.age=age

    def __str__(self) -> str:
        return f"{self.name}, {self.age} lat"
person=Person("Alicja",33)
print(str(person))
```

Obsługa wyjątków:

Zrozumienie mechanizmu wyjątków w Pythonie.

Stosowanie bloków try, except, finally do obsługi błędów i wyjątków.

```
try:#spróbuj wykonać ten kod
    instrukcje
except blad:#jeśli wystąpi błąd blad wykonaj instrykcje2
    instrukcje2
except blad:#jeśli wystąpi błąd blad wykonaj instrykcje3
    instrukcje3
finally:#jeśli nadal inny błąd wykonaj instrukcje4
    instrukcje4
```

Praca z plikami:

Otwieranie, czytanie, zapisywanie i zamykanie plików.

Rozumienie różnych trybów dostępu do plików (np. odczyt, zapis).

`r` - odczyt `w`- zapis , nadpisując dane w pliku `a`- zapis, dodając dane do pliku

```
with open(nazwa_pliku.txt, ('r'/'w'/'a')) as file:
    file.read()
    for linia in file:
        linia.strip()
    file.write()
file.close()
```

Wprowadzenie do testowania kodu:

Zrozumienie znaczenia testowania, podstawowe testy jednostkowe.

`pip list` -lista zainstalowanych biblioek

`pip freeze` -lista zainstalowanych biblioek z wersjami

`pip freeze > requirements.txt` -lista zainstalowanych biblioek z wersjami wrzuca do pliku txt

`pip install -r requirements.txt` -instaluje wszystkie biblioteki z pliku txt

TESTOWANIE

`python -m venv test` -tworzy środowisko o nazwie "test", izoluje środowisko dla danego pliku

`source [nazwa środowiska ]/Scripts/activate` -uruchomienie środowiska

`deactivate` -deaktywuje srodowisko testowe

`pip unistall -r requirements.txt` -odinstalowuje biblioteki tylko dla środowiska testowego

`pip install -r requirements.txt` -instalowuje biblioteki tylko dla środowiska testowego

`python -m pytest [nazwaplikju do testowania]` -uruchomienie testu

```
from car_class_logging import Car
from car_class_logging import ElektricalCar
import pytest

@pytest.fixture
def get_car():
    return Car("Toyota","Yaris",2023)

@pytest.fixture
def get_electric_car():
    return ElektricalCar("Tesla","Model 3",2024,75)

def test_car(get_car):
    expected_output="Marka: Toyota\nModel: Yaris\nRocznik: 2023\nProdukcja: nie"
    assert get_car.display_info()==expected_output

def test_car_start_production(get_car):
    get_car.start_production()
    assert get_car.is_production is True
```

Dobre praktyki programistyczne:

Stosowanie komentarzy, dokumentowania kodu, czytelność kodu, przestrzeganie konwencji nazewnictwa.

best practis: https://peps.python.org/pep-0008/

konwencje nazewnictwa

CamelCase -klasy

snake_case -zmienne